Maximal Utility Rate Allocation for Energy Harvesting Wireless Sensor Networks

Bo Zhang Robert Simon Hakan Aydin
Department of Computer Science
George Mason University
Fairfax, Virginia, USA, 22030
{ bzhang3, simon, aydin }@cs.gmu.edu

ABSTRACT

There is currently tremendous interest in deploying energy-harvesting wireless sensor networks. Engineering such systems requires striking a careful balance between sensing performance and energy management. Our work addresses this problem through the design and analysis of a harvesting-aware utility-based sensing rate allocation algorithm. Based on a network utility formulation, we show that our algorithm is optimal in terms of assigning rates to individual nodes to maximize overall utility, while ensuring energy-neutral operation. To our knowledge, our work is the first optimal solution that maximizes network utility through rate assignments for tree-structured energy harvesting sensor networks. Our algorithm is fast and efficient with running time $O(N^3)$, where N is the number of nodes. We evaluate the performance, scalability, and overhead of our algorithm for various utility functions and network sizes, underlining its significant advantages.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Distributed networks; C.2.2 [**Network Protocols**]: [Applications]

General Terms

Algorithms, Performance, Experimentation

Keywords

Sensor networks, Energy harvesting, Rate allocation

1. INTRODUCTION

The rapid introduction of new software and hardware functionalities has stimulated the development of complex wireless sensor network (WSN) applications. For this new generation of WSN applications, maximizing the value or *utility* of sensed data, as perceived by end-users, is of paramount importance. For most WSN systems, a major constraint to utility maximization is the limited energy availability of sensor nodes. One solution to this problem is to devise energy harvesting techniques to power sensor nodes at run-time [2, 4, 5]. Since the availability of environmental energy sources is often limited and highly time-varying, the goal of simple utility maximization is constrained by the need for managing unpredictable energy supplies. Our work addresses this problem by designing and analyzing a WSN-specific, energy-harvesting aware utility maximization algorithm.

Much previous work in this area assumes that application utility increases linearly with the rate at which sensor nodes sense or collect data [1, 11, 12, 13]; therefore the maximization of the network-wide data collection rate leads to utility maximization. However, for many WSN applications increasing the level of sensed data reporting only increases the utility of the application in sub-linear fashion. Consider, for instance, video or motion sampling as part of an intrusion detection system. Since humans can only move at a certain speed, sampling above a specific threshold only marginally increases the utility of the application. For this reason, our work models utility as a non-decreasing concave function, such that its rate of increase (*marginal utility*) decreases as the sensing and reporting rate increases [6, 21, 22]. The utility perceived by application end-user is the aggregate utility achieved jointly by all the nodes in the network.

With energy harvesting equipment such as solar panels or wind generators, sensor nodes harvest power from environmental sources and potentially sustain their operation perpetually. In [5], the concept of perpetual operation is formally stated as the *energy neutral condition*. This means that sensor nodes must always maintain positive energy storage levels to avoid energy depletion and hence operation interruption. A further complexity arises in multi-hop WSNs, where nodes must forward data received from other nodes in addition to their locally generated data. Given limited energy supplies, control algorithms must balance the local and external sensing and communication rates. Favoring local rates over external rates may result in high local utility, but low network-wide utility, since the rates of other nodes are throttled by the high local rate.

To solve the above issues we propose MAX-UTILITY, an *epoch*-based rate allocation algorithm. MAX-UTILITY is designed to maximize total application utility, while regulating nodes energy consumption resulted from sensing and reporting activities to guarantee energy neutral operation for every node. In our work, 'rate' is equated with packet rate, which is directly tied to sensing rate. An epoch is a time interval during which the amount of power harvested

MSWiM'11, October 31–November 4, 2011, Miami, Florida, USA.

remains relatively stable and is reasonably predictable. The length of an epoch may range from several minutes to a couple of hours, depending on the energy source type. Our approach exploits the concavity of the utility function, and a special property of tree-based networks to allocate rates to nodes as evenly as possible for achieving utility maximization, while maintaining the minimum sensing rate required by the application and the available energy and data forwarding capacity of the nodes.

We formally prove the optimality of MAX-UTILITY in the sense of utility maximization. For a system with N nodes, our algorithm has a time complexity of $O(N^3)$. We also develop a distributed version of this algorithm for use in systems without central control points. *To the best of our knowledge, this is the first optimal solution to maximize general network utility through rate assignments to individual nodes in tree-structured sensor networks, while guaranteeing energy neutrality.* A limitation of our approach is that it applies only to tree-based WSNs. However, since trees are a common logical routing structure for WSN systems [1, 3, 8], we believe our algorithm is widely applicable. While the related studies in [6, 21] target general multi-hop networks and offer innovative solutions, in general, they do not guarantee optimality. Finally, using solar harvesting traces obtained over a year-long study in [17], we have evaluated MAX-UTILITY against an alternative algorithm, under a variety of experiment settings. Our results show that MAX-UTILITY delivers superior utility improvement while ensuring energy neutral operation for all the nodes.

2. BACKGROUND AND RELATED WORK

Rate allocation for wireless sensor networks has been explored in [1, 6, 12, 13, 22]. In [1], the authors propose to maximize lexicographic rate assignments to sensor nodes. The rate assignment problem is formulated as linear optimization problem, and solved optimally by centralized and distributed algorithms. [12] proposes a rate control approach for a single energy harvesting node to achieve a series of objectives including the maximization of average sensing rate over time. These objectives are formulated as optimization problems and solved using multi-parametric control algorithms. [13, 14] propose a flow control algorithm for energy harvesting WSNs. [14] proposes an energy budgeting algorithm that defines the amount of energy a node can use for each epoch. The derived energy budget assignment is optimal in the sense that the variance of energy assigned across epochs is minimized. Using this formulation the flow control algorithm in [13] maximizes the amount of data collected over the network, given that no node consumes more energy than the assigned budget.

An implicit assumption in [1, 12, 13] is that the system utility increases linearly with the rate of nodes. However, for many applications the increase of utility slows down as rate increases (the *diminishing returns* principle). Using this observation, [6, 21, 22] model system utility as a concave and non-decreasing function of rate, and propose primal-dual based algorithms to maximize utility over the entire network. Among these works, [6] is the most related one to ours that target utility maximization for energy harvesting WSNs. However, [6, 21, 22] all assume specific utility functions that are continuously differentiable, which may limit their application for a more general class of functions. Moreover, their proposed solutions are not optimal, and can incur high control overhead and unpredictable running time, thus potentially limiting their practical implementation within resource-constrained WSN systems.

Research presented in [7, 15] formulate the Network Utility Maximization (NUM) problem for Internet congestion control. Primal-dual based algorithms are proposed to solve the problem. The issue we target shares the same structure with the NUM problem, but is for a WSN environment. The utility maximization problem has been also extensively studied for real-time embedded systems. [18] addresses utility maximization for energy-constrained systems that execute periodic real time tasks. The objective is to maximize the total utility obtained from execution of these tasks, while satisfying the deadlines of all the tasks, the tight system energy budget and the minimum system performance requirement. In [19], the authors extend the approach in [18] for solar-powered embedded systems. However, [19] only considers two different epochs in each day. [9, 10] address a similar problem as [19] but with a more complex system model. They assume a highly dynamic energy harvesting model, and assume that WSN applications have multiple discrete service and performance levels, and each has different utility values and energy demands. The problem is then to decide when to select which service level to run in order to maximize the total utility, without over-using the available energy. Finally, in [23], we propose energy management approach for maximizing the energy storage level of nodes in energy harvesting WSNs. The approach utilizes two energy saving techniques, dynamic voltage scaling and dynamic modulation scaling.

3. SYSTEM ARCHITECTURE

3.1 Environmental energy harvesting model

We assume each sensor node consists of an energy harvester head, and several energy consuming hardware units, including a CPU, a wireless transceiver, and required sensor suites. The harvester head is energy source-specific, such as a solar panel or wind generator. The energy storage unit, such as a rechargeable battery or super-capacitor, has a maximum energy capacity of Γ^{max} joules. This unit receives power from the energy harvester, and delivers power to the sensor node. We take the commonly used approach that the amount of harvested power is uncontrollable, but reasonably predictable, based on the source type and harvesting history [5, 6, 12]. To capture the time-varying nature of environmental energy, time is divided into *epochs* of length S. Harvested power is modeled as an epoch-varying function denoted by P_k^h, where k is the epoch number. P_k^h remains within each epoch k, but changes for different epochs. The time unit used for harvesting prediction is therefore one epoch. The *prediction horizon*, H is an interval containing a number of epochs during which harvesting predictions can be reasonably made. The length of H may vary depending on the type of environmental sources and prediction techniques used. Our approach does not depend on the length of H, and requires harvesting prediction for only the coming epoch.

3.2 Network and application model

We consider general WSN applications that periodically collect data from N sensor nodes populated over the target environment. The nodes are organized into a data collec-

tion tree using any tree construction protocol, such as the Collection Tree Protocol (CTP) [3]. A sensor node is denoted as V_i, and the base station is denoted as BS. Within a tree-based routing structure, at any time each node V_i is connected to BS by a single path ρ_i consisting of zero or more intermediate nodes. We use the notation $V_i \in \rho_j$ to indicate that V_i resides on a given path ρ_j.

A node V_i senses the environment and sends the resulting data in packets towards BS along the path ρ_i, at rate r_i. These packets are referred as *internal packets*. Each node V_i may also forward *external packets* at a certain rate. These are received from the set of descendant nodes $\{V_j\}$, where V_i resides on V_j's path to BS, i.e. $V_i \in \rho_j$. Therefore the total outbound traffic at V_i equals $r_i + \sum_{j:V_i \in \rho_j} r_j$. We refer to r_i as V_i's *internal packet rate*, and $\sum_{j:V_i \in \rho_j} r_j$ as its *external packet rate*.

Sensor nodes produce utility by sensing and reporting data to BS. Therefore we define the utility accrued by any node V_i as a function of its packet rate r_i, $U(r_i)$. U is a positive, non-decreasing and concave function. Then we define the aggregate utility U^{tot} accrued jointly by all N nodes in the network as:

$$U^{tot} = \sum_{i=1}^{N} U(r_i) \tag{1}$$

Our energy model assumes, without loss of generality, that sensing and processing energy costs are fixed, relatively negligible and can be ignored. We therefore consider the radio transceiver as the main sink for energy consumption. Packet routing results in two types of energy consuming activities, packet transmission and reception. We denote the energy spent on transmitting an internal or external packet as e^{tx}, where e^{tx} can be measured in advance. Packet reception also consumes energy. Our architecture assumes that reception energy is controlled by the MAC layer, using techniques such as TDMA or duty-cycled LPL approaches such as B-MAC [16]. Therefore we model per-epoch reception energy as a constant E^{rx} which is dependent on the epoch length. We can calculate, at each node V_i, the total energy consumed E_i^c by handling internal and external packets in an epoch of length S as:

$$E_i^c = e^{tx} \cdot (r_i + \sum_{j:V_i \in \rho_j} r_j) \cdot S + E^{rx} \tag{2}$$

We also assume that each node V_i has a pre-assigned per-epoch energy budget B_i. Such energy budgets can be produced by algorithms such as [9, 13, 14], based on the predictions of harvested energy for any of the epochs in the horizon. Further, related work such as [5, 6] require that a sensor node can consume no more than the amount of energy harvested in any epoch. For these models a node's allocated energy budget in any epoch is set to the amount of energy it can harvest. Any of these energy budgeting algorithms guarantees the energy-neutral operation across all epochs.

4. THE MAXIMAL UTILITY RATE ALLOCATION PROBLEM

In this section, we define our rate allocation problem for energy harvesting WSN systems. Our objective is to maximize the network utility U^{tot} (Eq. (1)), given the limited energy harvesting ability of nodes. We achieve this goal by adjusting packet rate r_i. Since the harvested power changes from epoch to epoch, the rates of nodes need to be re-adjusted in every epoch. We formulate this objective as an optimization problem called *Network Utility Maximization with Energy Harvesting* (*NUM-EH*) as:

$$\begin{aligned} Max \quad & U^{tot} && (3)\\ s.t. \quad & \forall V_i, \\ & E_i^c \le B_i && (4)\\ & r_i \ge r^{min} && (5)\\ & r_i + \sum_{j:V_i \in \rho_j} r_j \le R_i^{cap} && (6) \end{aligned}$$

The optimal solution to the above problem consists of the rates for all N nodes, i.e. $\{r_1, \ldots, r_N\}$ that maximizes U^{tot}. The constraint (4) enforces that the energy consumption E_i^c of any node V_i must be smaller than the assigned energy budget B_i in an epoch. The constraint (5) enforces that the rate of any node must be higher than the minimum required value r^{min}, in order to maintain the basic service level at individual nodes. Finally, to avoid packet congestion the total packet rate at any V_i, i.e. $r_i + \sum_{j:V_i \in \rho_j} r_j$ must be smaller than V_i's packet forwarding capacity denoted by R_i^{cap}.

We observe that *NUM-EH* is a concave maximization problem with three linear constraints. This is seen from Eq. (2), where E_i^c is a linear function of r_i and the set of external rates $\{r_j\}$. We notice that this problem is a special case of the well-known network utilization maximization problem [6, 7, 15] which can be solved using primal-dual based algorithms. However, such algorithms are typically too computationally expensive for resource-constrained sensor nodes. We instead propose a polynomial-time algorithm to solve the problem optimally and cost-effectively.

5. RATE ALLOCATION ALGORITHM

In this section, we propose the algorithm MAX-UTILITY that optimally solves problem *NUM-EH*. MAX-UTILITY is applicable to arbitrary utility functions that are concave and non-decreasing. We first propose a centralized version of this algorithm that can be run on a base station. We then show how to implement MAX-UTILITY in a fully distributed fashion so resource-constrained sensor nodes can collaboratively produce optimal rate assignments.

First, we show that the constraint (4) and (6) can be combined into one single constraint as follows. Substituting the equality (2) into constraint (4) gives:

$$E_i^c = e^{tx} \cdot (r_i + \sum_{j:V_i \in \rho_j} r_j) \cdot S + E^{rx} \le B_i \tag{7}$$

yielding:

$$r_i + \sum_{j:V_i \in \rho_j} r_j \le \frac{B_i - E^{rx}}{e^{tx} \cdot S} \tag{8}$$

The right-hand side of the above inequality, $\frac{B_i - E^{rx}}{e^{tx} \cdot S}$, is a constant as B_i, E^{rx}, e^{tx} and S are all known constants. This can be combined with constraint (6) into one single constraint, $r_i + \sum_{j:V_i \in \rho_j} r_j \le CAPACITY_i$ where:

$$CAPACITY_i = Min \left\{ R_i^{cap}, \frac{B_i - E^{rx}}{e^{tx} \cdot S} \right\} \tag{9}$$

We refer to $CAPACITY_i$ as the *rate capacity* of node V_i. We can re-write problem *NUM-EH* concisely as:

$$Max \quad U^{tot} = \sum_{i=1}^{N} U(r_i)$$

$$s.t. \quad \forall V_i, r_i \geq r^{min} \quad (10)$$

$$\forall V_i, r_i + \sum_{j:V_i \in \rho_j} r_j \leq CAPACITY_i \quad (11)$$

We refer to constraint (10) as the *min-rate constraint*, and constraint (11) as the *capacity constraint*.

5.1 The centralized version

Now we present the centralized version of algorithm MAX-UTILITY. Before the algorithm starts, BS collects two pieces of information from each node. The first is the rate capacity $CAPACITY_i$, computed locally by each node using Eq. (9). The second is the node id of parent of each node, and the BS uses the parenthood relation of nodes to derive the structure of the existing data collection tree. MAX-UTILITY allocates rates as evenly as possible to nodes, while also satisfying constraints (10-11). Given an arbitrary concave, non-decreasing functions U, this will maximize network utility U^{tot}. This property is formally given later in Proposition 1. MAX-UTILITY runs in multiple iterations, and assigns rates to a subset of nodes in each iteration. The iteration ends when rates are assigned to all N nodes.

$CAP[i]$	The remaining rate capacity of node V_i
$unassigned[i]$	The set of unassigned nodes in V_i's subtree τ_i
$\|unassigned[i]\|$	The size of $unassigned[i]$
$ASSIGNED\_SET$	The set of assigned nodes
$\tilde{R}[]$	The rate assignment derived by MAX-UTILITY
$r_c[i]$	The maximum common rate for nodes in $unassigned[i]$
V_u	The node with the least $r_c[i]$ among all the unassigned nodes

Table 1: List of notations

The algorithm has two lists of inputs and one output. The first input list contains N rate capacities $CAPACITY_i$, one for each node V_i. The second list contains N vectors, one for each V_i. Denote the subtree rooted at node V_i by τ_i. The vector of each V_i contains all the nodes in the subtree τ_i rooted at V_i. This vector is derived based on the previously discovered tree structure. For notational simplicity we re-use τ_i to represent this vector. The output is the rate assignment vector derived by MAX-UTILITY, denoted by $\tilde{R}[]$.

MAX-UTILITY uses one global variable and three per-node variables that are updated from iteration to iteration. The global variable $ASSIGNED\_SET$ is a set containing all the nodes in the network that have been assigned rates so far. As will be seen later, in each iteration MAX-UTILITY adds at least one node to $ASSIGNED\_SET$. The algorithm terminates when all N nodes are in $ASSIGNED\_SET$. Next we declare three per-node variables. First, $CAP[i]$ is the remaining capacity of node V_i, initialized to $CAPACITY_i$. Second, $unassigned[i]$ is a subset of τ_i containing any nodes in τ_i that have not yet been assigned rates. $unassigned[i]$ includes V_i itself, and is initialized to τ_i. Finally, $r_c[i]$ is the maximum common rate for nodes in $unassigned[i]$. Table 1 summarizes these notations. This construction is illustrated in Fig. 1. In the 1^{st} iteration, when none of nodes in subtree τ_7 rooted at V_7 have been assigned, $unassigned[7]$ contains all four nodes in τ_7, $\{V_1, V_2, V_6, V_7\}$. The last variable $r_c[i]$ is the maximum common rate that can be assigned to the nodes in $unassigned[i]$. $r_c[i]$ is computed by dividing V_i's remaining rate capacity $CAP[i]$ by $|unassigned[i]|$ which is the number of nodes in $unassigned[i]$, i.e. $r_c[i] = \frac{CAP[i]}{|unassigned[i]|}$. For example, in the 1^{st} iteration, $r_c[7] = \frac{CAPACITY_7}{|unassigned[7]|} = \frac{16}{4} = 4$.

Algorithm 1 MAX-UTILITY

```
1: - Input: {CAPACITY_i} and {τ_i}; Output: R̃[]
2: - Initialization: ASSIGNED_SET = ∅,
   ∀V_i, CAP[i] = CAPACITY_i, unassigned[i] = τ_i
3: for each node V_i in the network do
4:    If CAP[i] < r^min · |unassigned[i]|, then return ∅
5: end for
6: while |ASSIGNED_SET| < N do
7:    for any node V_i ∉ ASSIGNED_SET do
8:       r_c[i] = CAP[i]/|unassigned[i]|
9:    end for
10:   Find out V_u which has the least r_c[i] among any V_i ∉
      ASSIGNED_SET
11:   for any node V_i ∈ unassigned[u] do
12:      Set R̃[i] = r_c[u], and add V_i to ASSIGNED_SET
13:   end for
14:   for any node V_i ∉ ASSIGNED_SET do
15:      if V_u ∈ unassigned[i], i.e. V_i is V_u's ancestor then
16:         CAP[i] = CAP[i] − r_c[u] · |unassigned[u]|
17:      end if
18:      if V_u ∈ unassigned[i] then
19:         for any node V_j ∈ unassigned[u] do
20:            Remove V_j from unassigned[i].
21:         end for
22:      end if
23:   end for
24: end while
25: return R̃[].
```

MAX-UTILITY, shown in Algorithm 1, is specified as follows. Line 2 initializes $ASSIGNED\_SET$ to $\emptyset$, $CAP[i]$ to $CAPACITY_i$, and $unassigned[i]$ to τ_i. In lines 3-5, for any node V_i we check whether it has sufficient capacity to sustain the minimum required rate r^{min} for all the nodes in its subtree τ_i. If there exists a node with insufficient capacity, then we know there is no feasible rate assignment that can satisfy the min-rate constraint and capacity constraint (at V_i) at the same time. At this point MAX-UTILITY terminates immediately with empty $\tilde{R}[]$. Line 6 starts the rate assignment loop which will terminate when all N nodes are assigned. In lines 7-9, for any node $V_i \notin ASSIGNED\_SET$, we compute the maximum common rate $r_c[i]$ that can be assigned to the nodes in $unassigned[i]$ (line 8). In line 10, we find the node V_u which has the least $r_c[i]$ among any $V_i \notin ASSIGNED\_SET$. Then for any nodes in $unassigned[u]$, MAX-UTILITY assigns $r_c[u]$ to them and adds them to $ASSIGNED\_SET$ (line 11-13).

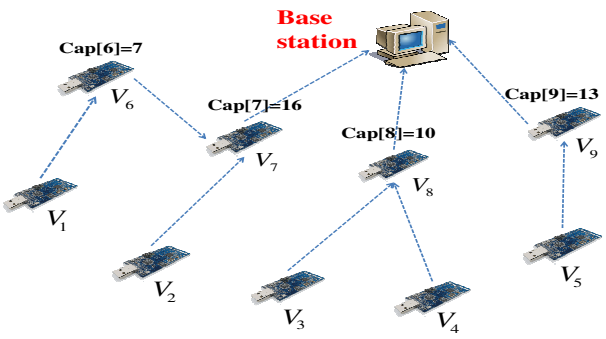

Figure 1: Example Data Collection Tree

Next, lines 14-23 update $CAP[i]$, and $unassigned[i]$ for any nodes without an assigned rate. Specifically, since any node V_i where $V_u \in unassigned[i]$ are ancestors of V_u and need to forward packets received from the nodes in $unassigned[u]$, we need to subtract $r_c[u] \cdot |unassigned[u]|$ (i.e. the total traffic that V_i received from V_u) from their $CAP[i]$ values (line 15-17). For any other nodes that are not ancestors of V_u, their $CAP[i]$ values remain the same. Moreover, any nodes $V_j \in unassign[u]$ with newly assigned rates must be removed from the $unassigned[i]$ sets of any node V_i that is an ancestors of V_u, i.e. $V_u \in unassigned[i]$ (line 18-22). The rate assignment process from line 7 to 23 continues until all N nodes are assigned.

As can be seen, in each iteration, MAX-UTILITY picks a V_u which has the least common rate $r_c[i]$ among any unassigned node V_i, and assigns $r_c[u]$ uniformly to any nodes in $unassigned[u]$, then produces a pruned tree by removing any newly assigned nodes. We refer to the node with the least common rate $r_c[i]$ among any nodes in a tree (or pruned tree) as the *critical node* of the tree. Note that in any iteration, the selected V_u is the critical node in the tree (pruned tree). The concept of critical node plays an important role in our optimality proof.

Steps of MAX-UTILITY are illustrated through an example. In Fig. 1, before the 1^{st} iteration starts, $r_c[6] = \frac{7}{2} = 3.5$, $r_c[7] = \frac{16}{4} = 4$, $r_c[8] = \frac{10}{3}$, $r_c[9] = \frac{13}{2} = 6.5$. Line 10 picks V_8 as V_u, and line 12 assigns $\tilde{R}[3] = \tilde{R}[4] = \tilde{R}[8] = \frac{10}{3}$, and adds V_3, V_4, V_8 to $ASSIGNED\_SET$. Since this assignment does not affect any other nodes, lines 14-23 are not executed. In the 2^{nd} iteration, V_6 is picked as V_u, so MAX-UTILITY assigns $\tilde{R}[1] = \tilde{R}[6] = 3.5$, and updates $CAP[7] = 16 - 7 = 9$, and $unassigned[7] = \{V_2, V_7\}$. V_1, V_6 are added to $ASSIGNED\_SET$. In the 3^{rd} iteration, V_7 is picked as V_u as $r_c[7] = 9/2 = 4.5 < r_c[9] = 6.5$. MAX-UTILITY assigns $\tilde{R}[2] = \tilde{R}[7] = 4.5$, marks V_2, V_7 assigned and makes no update. In the 4^{th} iteration, V_9 is picked as V_u, MAX-UTILITY assigns $\tilde{R}[5] = \tilde{R}[9] = 6.5$, and adds V_5, V_9 to $ASSIGNED\_SET$. Since all the nodes are now in $ASSIGNED\_SET$, the algorithm terminates and returns $\tilde{R}[] = \{3.5, 4.5, 10/3, 10/3, 6.5, 3.5, 4.5, 10/3, 6.5\}$.

5.2 Optimality of algorithm MAX-UTILITY

In Eq. (1), the network utility U^{tot} is defined as the aggregate utility accrued jointly by all the nodes, $\sum_{i=1}^{N} U(r_i)$. Since $U(r_i)$ is concave, U^{tot} can be maximized by assigning packet rates to nodes as evenly as possible, while satisfying the min-rate, and capacity constraints at any nodes. This observation is supported by the following proposition, which can be proved based on the properties of concave functions:

PROPOSITION 1. *Given n non-negative real numbers $\{r_1, \ldots, r_n\}$ where $\sum_{i=1}^{n} r_i \leq W$ (W is any non-negative constant), $\sum_{i=1}^{n} U(r_i)$ (U is a concave, non-decreasing function of r) is maximized when $r_1 = r_2 = \ldots = r_n = W/n$.*

Now we prove that the rate assignment $\tilde{R}[]$ derived by Algorithm MAX-UTILITY maximizes U^{tot}, while satisfying constraints (10-11). For the sake of contradiction, we assume that $\tilde{R}[]$ is not optimal, and that there exists an optimal assignment $R^*[]$ which differs from $\tilde{R}[]$. The network utility resulting from $R^*[]$ is denoted by $U^{tot,*}$. We will show that $R^*[] \neq \tilde{R}[]$ contradicts the optimality of $R^*[]$, hence proving that $R^*[i] = \tilde{R}[i]$ must hold for every node V_i in the network. We first propose Lemma 1, which is proved in the Appendix.

LEMMA 1. *Given an arbitrary tree-based network, the optimal rate assignment $R^*[]$ must assign equal rates to all nodes in $unassigned[u]$, where V_u is the critical node of the tree.*

Lemma 1 implies that $R^*[]$ must assign a common rate denoted as $R^*[u]$ to all the nodes in $unassigned[u]$. We now present Lemma 2, which shows that $R^*[u]$ must be equal to $r_c[u]$, i.e. the maximum common rate that can be assigned to nodes in $unassigned[u]$.

LEMMA 2. *Given an arbitrary tree-based network where V_u is the critical node in the tree, the optimal rate assignment $R^*[]$ must assign a common rate $r_c[u]$ to any nodes in $unassigned[u]$, regardless of the rates assigned to other nodes in the tree.*

The proof of Lemma 2 can be found in the Appendix. Since $\forall V_i \in unassigned[u], \tilde{R}[i] = r_c[u]$ holds (line 12), Lemma 2 implies that $\forall V_i \in unassigned[u]$, $R^*[i] = \tilde{R}[i]$ must hold. In other words $\tilde{R}[]$ derived by MAX-UTILITY is optimal for any nodes in $unassigned[u]$, where V_u is the critical node. Since assigning $R^*[u] = r_c[u]$ to any node $V_i \in unassigned[u]$ is optimal regardless of what rates are assigned to the rest of nodes, we can focus on finding the optimal rate assignment for a *partial tree* formed by disregarding any nodes in $unassigned[u]$ (i.e. removing nodes in $unassigned[u]$ from $unassigned[i]$ of any ancestors of V_u), and setting $CAP[i] = CAP[i] - r_c[u] \cdot |unassigned[u]|$ for any node V_i that is an ancestor of V_u. Note that the partial tree generated in this way contains exactly the set of nodes remaining unassigned at the beginning of the 2^{nd} iteration of MAX-UTILITY. Also each node in the partial tree must have the same $CAP[i]$ and $unassigned[i]$ as they have in the 2^{nd} iteration of MAX-UTILITY. Since Lemma 2 applies to an arbitrary tree, it holds for this partial tree as well. In other words, $R^*[]$ must assign a common rate $r_c[u']$ to any nodes $V_i \in unassigned[u']$, where $V_{u'}$ is the critical node of the partial tree.

Since the structure and capacity of the partial tree is the same as the updated tree in the 2^{nd} iteration, the critical node $V_{u'}$ must be exactly the V_u picked by MAX-UTILITY in the 2^{nd} iteration. Thus MAX-UTILITY must also assign $r_c[u']$ to any $V_i \in unassigned[u']$. Therefore, $\forall V_i \in$

$unassigned[u']$, $\tilde{R}[i] = r_c[u']$ must hold. Since $\forall V_i \in unassigned[u']$, $R^*[i] = r_c[u']$ holds as implied by Lemma 2, we have $\forall V_i \in unassigned[u']$, $\tilde{R}[i] = R^*[i]$ must hold. That is, $\tilde{R}[]$ is optimal for any nodes in $unassigned[u']$. Again, we may disregard any nodes in $unassigned[u']$ and focus on the new partial tree formed in this way. The new partial tree has the same structure and capacities as the updated tree in the 3^{rd} iteration of MAX-UTILITY. As we iteratively prune the tree (partial tree) and apply Lemma 2 to the generated partial trees until all the nodes are pruned except the root BS, we have proved that the $\tilde{R}[]$ derived by MAX-UTILITY is completely identical to $R^*[]$, and therefore $\tilde{R}$ is optimal for the entire original tree.

Complexity analysis:

We show that for N nodes, the centralized MAX-UTILITY runs in $O(N^3)$ time. We first focus on one iteration of the while-loop from lines 6-24. The for-loop from lines 7 to 9 has complexity $O(N)$, since there are $O(N)$ unassigned nodes in each iteration. The selection of V_u in line 10 has complexity $O(N)$ as this is equivalent to finding the minimum element among $O(N)$ $r_c[i]$ values. The rate assignment to nodes in $unassigned[u]$ in lines 11-13 has complexity $O(N)$ as well, since the size of $unassigned[u]$ is $O(N)$. The for-loop from lines 14 to 23 has complexity of $O(N^2)$. In each loop, we need to update $CAP[i]$ and $unassigned[i]$ for $O(N)$ unassigned nodes V_i. Updating $CAP[i]$ for each V_i takes $O(1)$ time (line 16), while updating each $unassigned[i]$ take $O(N)$ time as it requires removal of $O(N)$ nodes (lines 19-21). Thus, the entire for-loop completes in $O(N^2)$ time as there are $O(N)$ nodes $V_i \notin ASSIGNED\_SET$. Therefore, each iteration of MAX-UTILITY completes in $O(N^2)$ time.

Next the number of iterations of MAX-UTILITY is given by $O(N)$, as the set $ASSIGNED\_SET$ is initially empty and each iteration adds at least one node to $ASSIGNED\_SET$. Therefore, the total complexity of MAX-UTILITY is $O(N^3)$. For practical purposes we expect that the number of iterations will be small. This is because the nodes that are close to the root have more descendent nodes, hence much larger $unassigned[i]$ than the nodes in the lower levels of the tree. Therefore the high level nodes usually have small $r_c[i]$ values and are more likely to be picked as V_u. Since once these nodes are picked as V_u, all nodes in their $unassigned[i]$ sets are assigned and added to $ASSIGNED\_SET$ at once in one iteration, the set $ASSIGNED\_SET$ will grow rapidly and the expected number of iterations of MAX-UTILITY should be small. We will justify this assertion through simulation analysis.

5.3 The distributed version

We now present the distributed version of MAX-UTILITY, referred as MAX-UTILITY-D. We decompose and distribute the computations in Algorithm 1 to every node in the network. The purpose of MAX-UTILITY-D is to support systems that do not possess or make use of a single resource-rich central control point. MAX-UTILITY-D only requires a single coordinator node such as a routing tree root. The tree root can be any node in the network. The root does not perform computations but only disseminates the minimum common rate $r_c[u]$ to all other nodes.

MAX-UTILITY-D is specified as follows. Initially, each node individually computes its rate capacity $CAPACITY_i$ using Eq. (9). Then MAX-UTILITY-D starts an initialization stage during which all the nodes send an empty control packet to the root. If a node V_i receives a control packet from another node V_j, then V_i knows V_j resides in its subtree τ_i. By the end of the initialization stage, each node knows its subtree τ_i, and hence the initial $unassigned[i]$. Using $CAPACITY_i$ and $|unassigned[i]|$, V_i computes the common rate $r_c[i]$ that can be assigned to nodes in $unassigned[i]$.

Each iteration of MAX-UTILITY-D consists of two stages. The first stage determines the minimum common rate $r_c[u]$ and the critical node V_u in the tree by requiring all the nodes forward their $r_c[i]$ values to the root. In the second stage, the root disseminates $r_c[u]$ across the network, and all nodes in $unassign[u]$ receive and use $r_c[u]$ as their packet rate. In the first stage, each leaf node in the tree V_i sends its computed $r_c[i]$ towards the root. V_i's parent V_j receives $r_c[i]$, compares $r_c[i]$ to $r_c[j]$ of itself, and forwards the larger of the two upwards. V_j also temporarily stores any received $r_c[i]$. This process proceeds over the entire tree in bottom-up fashion, and finally the root discovers the minimum common rate $r_c[u]$ in the tree. Then in the second stage the root disseminates $r_c[u]$ across the tree. When a node receives $r_c[u]$ from the root, it compares $r_c[u]$ to its local r_c to identify whether it is the critical node. If it is not the critical node, then it compares $r_c[u]$ to each of the previously stored r_c values of its descendent nodes to find out who is the critical node. In this way the critical node V_u identifies itself, and commands all unassigned nodes in $unassigned[u]$ to use $r_c[u]$ as their rates for the coming epoch. This rate assignment command can be piggybacked in the packet for disseminating $r_c[u]$.

After the dissemination of $r_c[u]$ completes, all the nodes has learned the identity of the critical node V_u, either by comparing $r_c[u]$ to its own or stored r_c values, or from the rate assignment command issued by V_u. Finally, we need to update $CAP[i]$ and $unassigned[i]$ of the nodes remaining unassigned. If a node V_i is an ancestor of V_u, it has to update $CAP[i] = CAP[i] - r_c[u] \cdot |unassigned[u]|$ and then remove all nodes in $unassigned[u]$ from $unassigned[i]$. At the end of the iteration, each node discards the temporarily stored information and continues to the next iteration. In the next iteration, only the nodes remaining unassigned participate. The rate assignment iteration continues until all the nodes are assigned.

Complexity analysis:

In MAX-UTILITY-D, the tree root only disseminates $r_c[u]$. Each non-root node requires $O(N)$ comparisons in each iteration. This is because each non-root node compares its own r_c to $O(N)$ r_c values received from its descendent nodes (the 1^{st} stage), and compares $r_c[u]$ disseminated by the root to its own r_c and $O(N)$ temporarily stored r_c. Further, updating $CAP[i]$ and $unassigned[i]$ takes $O(N)$ time for each node as described in the complexity analysis of centralized MAX-UTILITY.

Now we analyze the message complexity of the algorithm. The initialization stage requires one round of network-wide data collection. In each iteration, the determination of $r_c[u]$ requires all the nodes to report their $r_c[i]$ values, thus incurring another round of data collection. The root then announces $r_c[u]$ to all the nodes, incurring one round of network-wide data dissemination. In each data collection and dissemination round, each node sends exactly one packet. Given $O(N)$ iterations, MAX-UTILITY-D needs $O(N)$ rounds of network-wide data collection and dissemination. Finally, as mentioned in the analysis of the centralized MAX-

UTILITY, the expected number of iterations is much lower than N in practice.

6. PERFORMANCE EVALUATION

Though we have formally proved the optimality of MAX-UTILITY, we have conducted a series of simulation experiments to evaluate its performance gain and overhead. We compare our algorithm against an alternative heuristic, called *Random Rate Augmentation* (RRA). The specification of RRA is given in the Appendix. By design, the rate assignment derived by RRA is also feasible in term of satisfying constraint (10-11). Although RRA is not optimal, it achieves reasonably high utility values as it maximizes total packet rate (flow) over the tree. Other algorithms, such as the one proposed in [6] are not directly comparable to ours, since their application and network model are different.

We compare the two rate allocation algorithms under various experiment settings that consider different energy budgeting schemes, utility functions, and network sizes. Without loss of generality, we consider solar-powered sensor networks in our simulation. The solar power harvesting profile is obtained from the Hamburg University of Technology [17]. We set the length of horizon to one day (24 hours) which is further divided into 96 epochs each with length of 15 minutes. We assume two energy budgeting schemes, denoted as Strictly Energy Neutral (SEN) and Maximum Uniform Budget (MUB). The SEN scheme assigns the exact amount of harvested energy as the energy budget for any epoch. For the evening epochs with extremely low harvested energy, SEN assigns a minimum energy budget equaling $5J$ to maintain the minimum rate r^{min}. The MUB scheme computes the maximum uniform energy budget across all 96 epochs while satisfying the min-rate and capacity constraints, and assigns this budget to every epochs. The MUB scheme uses the algorithm proposed in [9]. Note that the energy budget varies across epochs if SEN scheme is used, while the budget remains constant if MUB scheme is used. We are interested in evaluating how different budgeting schemes affect the network utility. This helps system designers in choosing the right energy budgeting scheme for their applications. We consider three different concave utility functions: $log(100r + 1)$ (denoted as LOG), $\sqrt{100r}$ (SQR) and $log[\sqrt{1000r} + 1]$ (denoted as composite utility or CPST). We use the coefficients 100 and 1000 with r because r is commonly small. Finally, we repeat our simulation in six networks of different size, containing 25, 36, 64, 100, 169, 225 nodes respectively. We examine the accrued utilities, and the algorithm overhead as the network grows.

The sensor nodes are organized into a collection tree using the CTP protocol. The energy consumption for transmitting a packet is randomly selected from the range $(0.05 - 0.1)J$. This range is obtained based on the measurements in [20]. The energy storage device has capacity of 1000J. The initial energy level in the horizon is set to 500J. The minimum rate r^{min} is set to 0.01, i.e. 1 packet per 100 seconds.

6.1 Simulation results

We evaluated algorithm MAX-UTILITY and RRA in TOSSIM/TinyOS. We present our evaluation results along four dimensions: accrued network utility U^{tot}; rate (r_i) distribution among nodes; energy level variation across epochs of two selected nodes; algorithm running time and control overhead. We calculate the energy level Γ_k of a node at the end of the k^{th} epoch using the formulas below:

$$\Gamma_1 = Min.\{\Gamma^{init} + P_1^h \cdot S - E_1^c, \Gamma^{max}\}$$
$$\Gamma_k = Min.\{\Gamma_{k-1} + P_k^h \cdot S - E_k^c, \Gamma^{max}\}$$

Γ^{init} is the initial energy level. Γ^{max} is the energy capacity. Recall that P_k^h is the harvested power, S is the epoch length, and E_k^c is the energy consumed in epoch k (Eq. (2)). Γ_{k-1} is the ending energy level in the previous epoch $k-1$, which is also the starting energy level in epoch k. $P_k^h \cdot S$ gives the energy harvested in epoch k.

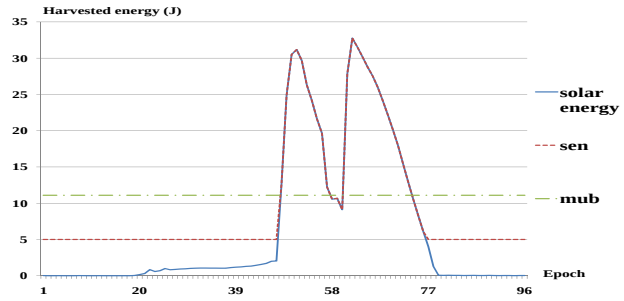

Figure 2: Solar energy harvested throughout a day.

In Fig. 2, we show the solar energy trace, along with the energy budget assignments derived by SEN and MUB schemes. As seen from the figure, the amount of solar energy reaches its peak in the afternoon due to high sunlight intensity. The line representing the energy budget assignment derived by SEN overlaps the line representing the solar energy in noon and afternoon. This is because SEN assigns the amount of harvested energy as the energy budget for any epochs, except for the evening epochs in which the harvested energy is less than 5J. Finally, MUB assigns the highest uniform budget (around 11J) for any epochs.

6.2 Network utility

In Fig. 3 and 4, we compare the utility accrued by MAX-UTILITY and RRA algorithm. We show the results for the 100-node network. In each figure we plot the utility values in all 96 epochs, for all three utility functions. In each figure, we plot six utility lines, one for each combination of the three utility functions and two rate allocation algorithms.

Fig. 3 assumes SEN energy budgeting scheme. Since SEN scheme assigns the harvested energy as the energy budget, we observe that the variations in accrued utility indicate similar pattern as the harvested energy (Fig. 2). Specifically the utility reaches its peak in the afternoon since high solar energy availability leads to high rate capacity $CAPACITY_i$ at any nodes. For any utility functions, MAX-UTILITY (OPT) achieves significantly higher utility than RRA. Specifically for SQR, the utility value increases above 500 in the afternoon if MAX-UTILITY is used. However, if RRA is used, the utility accrued never increases above 300. For the rest of the day, MAX-UTILITY achieves utility figures that are mostly above 200, while RRA achieves utility figures that are around 100.

In Fig. 4, we repeat the simulation for MUB energy budgeting. Because the MUB scheme is used, the energy budget B is uniform across different epochs, thus yielding constant rate capacities at nodes and much more stable utility per-

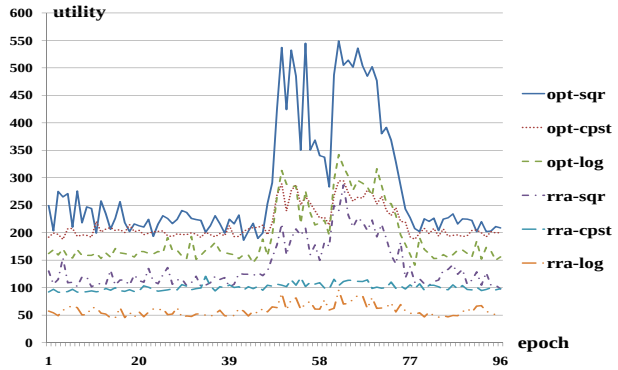

Figure 3: Utility accrued by max-utility (opt) and rra, with different utility functions, SEN budgeting.

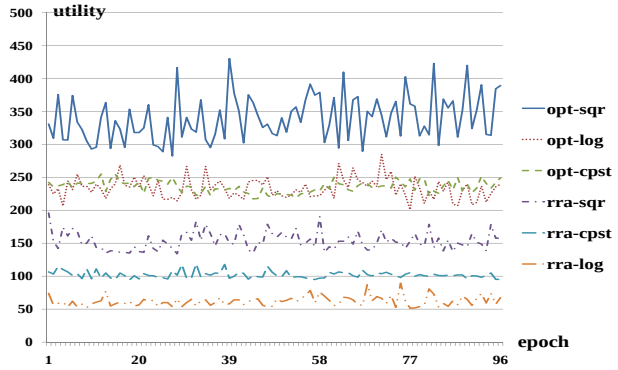

Figure 4: Utility accrued by max-utility (opt) and rra, with different utility functions, MUB budgeting.

formance in all six lines. Again, MAX-UTILITY achieves much higher utility than RRA. For SQR, the utilities accrued by MAX-UTILITY remain around 350 throughout the day, while the one for RRA remains around 150. As seen from Fig. 3 and 4, we observe that the MUB scheme maintains stable and high utility at any time and is hence suitable for the applications which desire utility stabilization. On the other hand, SEN energy allocation achieves very high utility when harvesting ability is high and is suitable for the applications where maximum utility is sought.

6.3 Rate assignment and energy storage level

In this section, we show the resulting rate assignment and energy storage levels of the two algorithms. First, we plot in Fig. 5 the distribution of rates of all the nodes in the 100 nodes network at midnight, while using the SEN scheme and the CPST utility function. From the figure, we observe that MAX-UTILITY tends to distribute rates more evenly across nodes than RRA. The rate assignment by MAX-UTILITY has standard deviation of 0.1268, while for RRA the deviation equals 0.1936. In Fig. 6(a-b) we plot the energy levels of a heavily-loaded node and a lightly-loaded node in 96 epochs. The heavily-loaded node is high in the tree, thus has higher workload and energy demand and lower energy

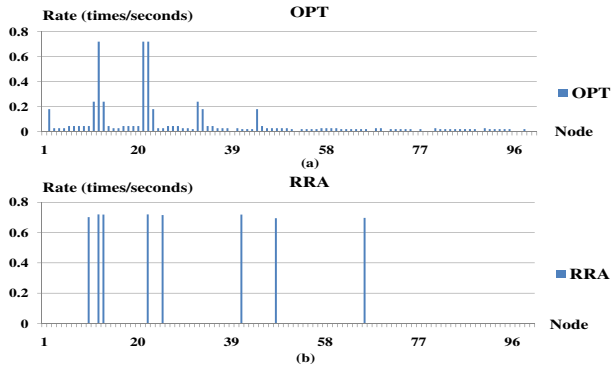

Figure 5: Distribution of rate (times/seconds) of all the nodes at midnight by (a) MAX-UTILITY; (b) RRA.

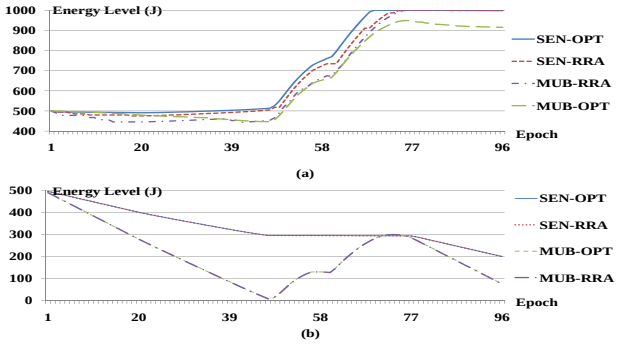

Figure 6: Variation of Energy levels over 96 epochs of (a) a lightly-loaded node; (b) a heavily-loaded node.

level. As seen from the figure, the energy neutral condition is maintained for both nodes at all times.

6.4 The impact of network size

Now we study the impact of network size on performance and overhead of MAX-UTILITY. In Fig. 7, we increase the size of the network from 25 nodes to 225 nodes, and plot the utility accrued by MAX-UTILITY. This simulation assumes SEN energy budgeting and LOG utility. We observe from the plot that the utility grows steadily as the network grows from 25 nodes to 169 nodes. As computed, the average utility is 78.10 for 25 nodes network, 109.34 for 36-nodes network, 142.84 for 64-nodes network, 203.79 for 100-nodes network, 396.54 for 169-nodes network, respectively. However, as the network size increases to 225, the utility increases to only 403.48. In other words the utility growth almost stops. This is because that, although there are potentially more utility contributors in the 225-nodes network, the achievable utility is constrained by the nodes that are close to the tree root, since the rate capacity of these nodes does not change with the network size. System designers must be aware of this phenomenon while choosing the network size and routing patterns.

We measure the actual running time of MAX-UTILITY in terms of the number of iterations shown in Table 2. Again, we assume 6 network sizes. As seen from the table, the

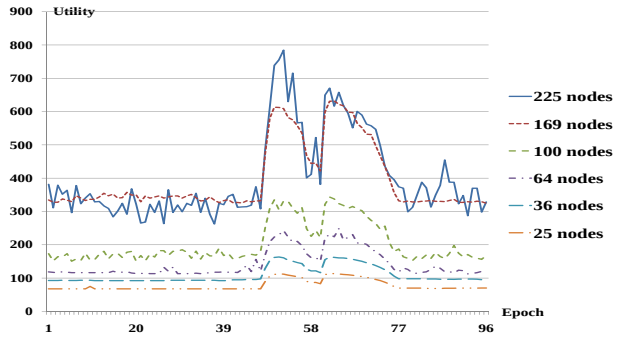

Figure 7: Total utility accrued by MAX-UTILITY in networks of different size

number of iterations does not increase with the network size. On the contrary, we observe that the 25-node network requires the largest number of iterations, 9.97, while the 64-nodes network requires the smallest number of iterations, 6.99. Based on the tree structure we observe that the number of iterations depends heavily on the number of children the tree root has. As mentioned in 5.2, the children of the root have large group of descendent nodes, hence are more likely to be picked as the critical node V_u. Once they become V_u, all their descendent nodes are added to the set $ASSIGNED\_SET$ in one iteration. This leads to the rapid growth of $ASSIGNED\_SET$.

Finally, we count the average number of control packets sent by a node in each invocation of the distributed MAX-UTILITY to be 27.32 in 100-nodes network. If the node operates at rate $r_i = 1$ packet per second then 900 data packets will be sent in an epoch. The packet overhead is then $27.32/900 \approx 3\%$. This indicates that the MAX-UTILITY-D is an entirely feasible alternative to MAX-UTILITY.

Num. of nodes	25	36	64	100	169	225
Num. of iterations	9.97	7.99	6.99	7.99	8.99	8.99

Table 2: The number of iterations of MAX-UTILITY as a function of network size

7. CONCLUSIONS

This paper addressed a utility maximization problem for energy harvesting sensor networks. Utility is defined as concave and non-decreasing function of nodes sensing rate. We proposed MAX-UTILITY, a rate allocation algorithm to maximize the total utility achieved jointly by all the nodes, while ensuring energy neutrality for any nodes. We formally proved the optimality of the algorithm, and indicated that the algorithm can derive optimal rate assignment in $O(N^3)$ time where N is the network size. Finally, extensive simulation results demonstrate its superior performance.

Acknowledgments: This work was supported, in part, by US National Science Foundation awards CNS-1016855 and CNS-0546244 (CAREER Award).

8. REFERENCES

[1] K.-W. Fan, Z. Zheng, and P. Sinha. Steady and fair rate allocation for rechargeable sensors in perpetual sensor networks. In *the 6th ACM conference on Embedded network sensor systems*, (SenSys'08), Raleigh, North Carolina, 2008.

[2] S. Farhan, S. Devyani, and P. Chou. Everlast: Long-life, supercapacitor-operated wireless sensor node. In *the 6th ACM conference on Embedded network sensor systems*, (Sensys'09), San Diego, California, 2009.

[3] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. In *the 6th ACM conference on Embedded network sensor systems*, (Sensys'05), Berkeley, California, 2009.

[4] X. Jiang, J. Polastre, and D. Culler. Perpetual environmentally powered sensor networks. In *the 4th ACM/IEEE international symposium on Information processing in sensor networks*, (IPSN'05), Los Angeles, California, 2005.

[5] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava. Power management in energy harvesting sensor networks. *ACM Trans. on Embedded Computing System*, 6(4):32, 2007.

[6] R. Liu, P. Sinha, and C. E. Koksal. Joint energy management and resource allocation in rechargeable sensor networks. In *the 29th IEEE Conference on Information Cmmunications*, (Infocom'10), San Diego, California, 2010.

[7] S. H. Low. A duality model of tcp and queue management algorithms. *IEEE/ACM Trans. on Networking*, 11:525–536, 2002.

[8] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. Tag: A tiny aggregation service for ad-hoc sensor networks. In *the USENIX Symposium on Operating Systems Design and Implementation*, (OSDI'02), Boston, MA, 2002.

[9] C. Moser, J. Chen, and L. Thiele. Reward maximization for embedded systems with renewable energies. In *the 14th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, (RTCSA'08), Kaohsiung, Taiwan, 2008.

[10] C. Moser, J.-J. Chen, and L. Thiele. Power management in energy harvesting embedded systems with discrete service levels. In *the 14th ACM/IEEE International Symposium on Low Power Electronics and Design*, (ISLPED'09), San Fancisco, California, 2009.

[11] C. Moser, L. Thiele, D. Brunelli, and L. Benini. Robust and low complexity rate control for solar powered sensors. In *the conference on Design, automation and test in Europe*, (Date'08), Munich, Germany, 2008.

[12] C. Moser, L. Thiele, D. Brunelli, and L. Benini. Adaptive power management for environmentally powered systems. *IEEE Transactions on Computers*, 59(4):478 – 491, 2010.

[13] D. K. Noh and K. Kang. A practical flow control scheme considering optimal energy allocation in solar-powered wsns. In *the 18th IEEE International Conference on Computer Communications and Networks*, (ICCCN'09), Washington, DC, 2009.

[14] D. K. Noh, L. Wang, Y. Yang, H. K. Le, and T. Abdelzaher. Minimum variance energy allocation for a solar-powered sensor system. In *the 5th IEEE International Conference on Distributed Computing in Sensor Systems*, (DCOSS'09), Marina Del Rey, California, 2009.

[15] D. P. Palomar and M. Chiang. A tutorial on decomposition methods for network utility maximization. *IEEE J. Sel. Areas Commun*, 24:1439–1451, 2006.

[16] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *the 2nd ACM International conference on Embedded networked sensor systems*, (SenSys'04), Baltimore, Maryland, 2004.

[17] C. Renner, J. Jessen, and V. Turau. Lifetime prediction for supercapacitor-powered wireless sensor nodes. In *the 8th GI/ITG KuVS Fachgespräch "Drahtlose Sensornetze"*, Hamburg, Germany, 2009.

[18] C. Rusu, R. Melhem, and D. Mossé. Maximizing the system value while satisfying time and energy constraints. In *the 23rd IEEE Real-Time Systems Symposium*, (RTSS'02), Washington, DC, 2002.

[19] C. Rusu, R. Melhem, and D. Mossé. Multi-version scheduling in rechargeable energy-aware real-time systems. *J. Embedded Comput.*, 1(2):271–283, 2005.

[20] V. Shnayder, M. Hempstead, B. rong Chen, G. W. Allen, and M. Welsh. Simulating the power consumption of large-scale sensor network applications. In *the 6th ACM conference on Embedded network sensor systems*, (Sensys'04), Baltimore, Maryland, 2004.

[21] L. Su, Y. Gao, Y. Yang, and G. Cao. Towards optimal rate allocation for data aggregation in wireless sensor networks. In *the 12th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, (Mobihoc'11), Paris, France, 2011.

[22] X. Wang and K. Kar. Cross-layer rate control for end-to-end proportional fairness in wireless networks with random access. In *the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, (Mobihoc'05), Urbana-Champaign, Illinois, 2005.

[23] B. Zhang, R. Simon, and H. Aydin. Energy management for time-critical energy harvesting wireless sensor networks. In *the 12th International Symposium on Stabilization, Safety and Security of Distributed Systems*, (SSS'10), New York City, New York, 2010.

APPENDIX

Proof of Lemma 1:

For the purpose of contradiction, we assume that, in $R^*[]$, nodes in $unassigned[u]$ are assigned unequal rates $\{\mu_1, \ldots, \mu_{|unassigned[u]|}\}$. However, as shown by Proposition 1 if this assumption holds we can always further increase $U^{tot,*}$ by equalizing $\{\mu_1, \ldots, \mu_{|unassigned[u]|}\}$ to a common rate $R^*[u]=\frac{\mu_1+\ldots+\mu_{|unassigned[u]|}}{|unassigned[u]|}$. This contradicts the optimality of $R^*[]$. Therefore, *equal rate assignment to nodes in* $unassigned[u]$ *is a necessary condition for the optimality of* $R^*[]$.

We still need to show such rate equalization does not violate the min-rate and capacity constraints (Eq. (10-11)). This is justified as follow. First, the min-rate constraint is not violated, because the new common rate $R^*[u]$ must be larger than the minimum of $\{\mu_1, \ldots, \mu_{|unassigned[u]|}\}$, which is in turn larger than r^{min} (due to the feasibility of $R^*[]$). The rate capacity constraint is also satisfied. This is because V_u is the critical node with the least common rate r_c among any nodes in the tree. Thus, $r_c[u]$ cannot be larger than $r_c[i]$ of any node V_i in $unassigned[u]$. Further, $R^*[u] \leq r_c[u]$ must hold, since if $R^*[u] > r_c[u]$ we have $R^*[u] \cdot |unassigned[u]| > r_c[u] \cdot |unassigned[u]|$, which gives $\mu_1 + \ldots + \mu_{|unassigned[u]|} > CAP[u]$. This implies that $R^*[]$ violates the capacity constraint which contradicts the feasibility of $R^*[]$. Since $R^*[u] \leq r_c[u]$, and $r_c[u] \leq r_c[i]$, $\forall V_i \in unassigned[u]$ we know the common rate $R^*[u]$ cannot be larger than the maximally allowed common rate $r_c[i]$ of any node V_i in $unassigned[u]$, hence cannot violate the capacity constraint at these nodes. Finally, the capacity violation cannot happen to any nodes outside $unassigned[u]$ since the total packet flow at V_u does not change after the rate equalization. Therefore, we have proved that in $R^*[]$, any nodes in $unassigned[u]$ must use a common rate.

Proof of Lemma 2:

In Lemma 1, we have shown that in $R^*[]$ all nodes in $unassigned[u]$ must be assigned a common rate $R^*[u]$, where V_u is the critical node in the tree. Now we show that this common rate $R^*[u]$ must be equal to $r_c[u]$ in order for $R^*[]$ to be optimal. Recall that $r_c[u] = \frac{CAP[u]}{|unassigned[u]|}$. Notice that nodes in $unassigned[u]$ cannot be assigned a common rate $R^*[u]$ that is higher than $r_c[u]$ as this will exceed V_u's capacity, so $R^*[u] > r_c[u]$ cannot hold. Therefore we assume $R^*[u] < r_c[u]$. We show this assumption cannot hold either, and thereby $R^*[u] = r_c[u]$ must hold.

If $R^*[u] < r_c[u]$ holds, we assert that the rate capacity of V_u is not fully utilized. This is because $R^*[u] < r_c[u]$ means $R^*[u] \cdot |unassigned[u]| < r_c[u] \cdot |unassigned[u]| = CAP[u]$ Since the left-hand side of the above inequality gives the total packet rate at V_u, if $R^*[u]$ is assigned to every nodes in $unassigned[u]$, this inequality implies V_u's capacity $CAP[i]$ is under-utilized.

Given that V_u's capacity is under-utilized, we make another assertion that there exists a node V_k, an ancestor of V_u, that must have its rate capacity fully utilized. This is because if all the ancestors of V_u are also under-utilized, then we can further increase $U^{tot,*}$ by increasing $R^*[u]$ by an amount of $\epsilon/|unassigned[u]|$, where $\epsilon > 0$ is the strictly positive, minimum un-utilized capacity among any ancestors of V_u. This contradicts the optimality of $R^*[]$. If there are multiple such V_k nodes, we consider the one that is in the lowest level of the tree.

Given the existence of V_k, we assert that in $R^*[]$ there must exist a node $V_j \in \tau_k$ and $V_j \notin unassigned[u]$ which has rate $R^*[j] > R^*[u]$. (Recall that τ_k is the subtree rooted at V_k.) This is because if no node in τ_k has rate higher than $R^*[u]$, then the total packet rate at V_k must be strictly smaller than $R^*[u] \cdot |\tau_k|$ ($|\tau_k|$ is the number of nodes in τ_k), and in turn no larger than $r_c[k] \cdot |\tau_k| = CAP[k]$, i.e. V_k's capacity. $R^*[u] \cdot |\tau_k| \leq r_c[k] \cdot |\tau_k|$ holds because $R^*[u] < r_c[u]$ is assumed, and $r_c[u] \leq r_c[k]$ since V_u has the least r_c among any nodes in the tree. This implies the capacity of V_k is under-utilized which contradicts our previous assertion that V_k is fully utilized. Thus we justify the existence of such V_j.

Next, given $R^*[j] > R^*[u]$, we know that the marginal utility of the concave utility function U at point $R^*[j]$ is lower than that at $R^*[u]$. Therefore we can further increase $U^{tot,*}$ by decreasing $R^*[j]$ by a strictly positive amount ϵ', and increasing $R^*[u]$ for each node in $unassigned[u]$ by the amount of $\frac{\epsilon'}{|unassigned[u]|}$, as long as $R^*[j] > R^*[u]$ holds. In other words, we can always increase $U^{tot,*}$ by reducing the difference between $R^*[j]$ and $R^*[u]$. This rate adjustment cannot violate the min-rate constraint as $R^*[u] > r^{min}$. The capacity constraint is also not violated, as the decrease of $R^*[j]$ does not increase the total packet rate at any nodes. The increase of $R^*[u]$ by $\frac{\epsilon'}{|unassigned[u]|}$ will not exceed the capacities of any V_u's ancestors inside τ_k, as long as ϵ' is no larger than the minimum unused capacity of any ancestors of V_u in τ_k. This is because V_k is the lowest ancestor of V_u whose capacity is fully utilized, hence any ancestors of V_u inside τ_k (i.e. below V_k) must have positive unused capacity. Further, the total packet rate at any V_u's ancestors outside τ_k does not change after the rate adjustment, so their capacity constraints cannot be violated either.

Therefore, the feasible increase of $U^{tot,*}$ contradicts the optimality of $R^*[]$, hence justifying that $R^*[u] < r_c[u]$ cannot hold, and $R^*[u] = r_c[u]$ must hold in order for $R^*[]$ to be optimal. Therefore we have proved Lemma 2.

Specification of the RRA algorithm

The RRA algorithm initially allocates r^{min} to every node, and updates the remaining capacity of nodes accordingly. If this violates the capacity constraints at some nodes, then we know there exists no feasible assignments for the given network. Otherwise, we start from that assignment and progressively improve U^{tot} by augmenting the rates of nodes with unused capacity. Specifically, based on the initial assignment, RRA randomly selects a node V_i with positive remaining capacity, and determines the maximum possible rate increment which V_i and its ancestors can afford. Then RRA adds up this increment to V_i's current rate, and zeros out its remaining capacity. RRA updates the remaining capacity of other nodes accordingly. This is the end of one round of augmentation. We then continue augmenting other nodes until no node has remaining capacity. The resulting rates of nodes is the final utility achieved by RRA.